Certified meshing of Radial Basis Function based isosurfaces

Chattopadhyay, A.; Plantinga, S.; Vegter, G.

Published in:
Visual computer

DOI:
10.1007/s00371-011-0627-2

Publication date:
2012

Vis Comput (2012) 28:445–462
DOI 10.1007/s00371-011-0627-2

ORIGINAL ARTICLE

Certified meshing of Radial Basis Function based isosurfaces

A. Chattopadhyay · S. Plantinga · G. Vegter

Published online: 9 September 2011

Abstract Radial Basis Functions are widely used in scattered data interpolation. The surface-reconstruction method using radial basis functions consists of two steps: (i) computing an interpolating implicit function the zero set of which contains the points in the data set, followed by (ii) extraction of isocurves or isosurfaces. In this paper we focus on the second step, generalizing the work on certified meshing of implicit surfaces based on interval arithmetic (Plantinga and Vegter in Visual Comput. 23:45–58, 2007). It turns out that interval arithmetic, and even the usually faster affine arithmetic, are far too slow in the context of RBF-based implicit surface meshing. We present optimized strategies giving acceptable running times and better space complexity, exploiting special properties of RBF-interpolants. We present pictures and timing results confirming the improved quality of these optimized strategies.

Keywords Certified meshing · Geometric computing · Radial Basis Functions · Interval arithmetic · Affine arithmetic

1 Introduction

RBF-based interpolants Radial Basis Functions (RBFs) provide a simple meshless method for the reconstruction of smooth geometric objects in the plane or in three-dimensional space from a finite point sample $v_1, \ldots, v_n$. The process consists of two steps: (i) computing an interpolating implicit function the zero set of which contains the sample points, followed by (ii) extraction of the isocurve or isosurface.

The radial basis interpolant constructed in step (i) is of the form

$$s(\underline{x}) = \sum_{k=1}^{n} w_k \, \varphi(\|\underline{x} - v_k\|) + p(\underline{x}), \tag{1}$$

where $\underline{x} \in \mathbb{R}^d$, for $d = 2, 3$, such that s is zero at the sample points (*centers*) v_k. To avoid obtaining the trivial zero solution, i.e. $s \equiv 0$, some of those v_ks are chosen from off-surface. Here p is a polynomial of low degree which depends on the particular choice of radial basis function, cf. [8], and $\|\underline{x} - v_k\|$ is the Euclidean distance between $\underline{x}$ and v_k. The radial basis function φ is a univariate function. Some popular RBFs are $\varphi(r) = r^3$ (triharmonic spline), $\varphi(r) = r^2 \log r$ (thin plate spline in 2D), $\varphi(r) = \sqrt{r^2 + c^2}$ (multiquadric), or $\varphi(r) = \exp(-r^2)$ (Gaussian). The *weights* w_k are determined by solving a system of interpolation equations. We omit the details, but refer to [3, 20] for further background.

The second step, namely isosurface extraction, is our main focus. In [14] we use *interval arithmetic (IA)* to extract regular level sets of a general smooth (C^1) implicit function. More precisely, the algorithm computes a piecewise linear surface, say S', which is isotopic to the actual zero set $S \equiv s^{-1}(0)$, and is guaranteed to have the same topology. Here, isotopy implies that there exists a continuous deformation between S and S' within the embedding space. The certified meshing algorithm [14] is akin to the Marching Cubes algorithm in the sense that it analyzes the topology of the isosurface on boxes in the plane or in space. If it cannot

A. Chattopadhyay (✉) · S. Plantinga · G. Vegter
University of Groningen, Groningen, The Netherlands
e-mail: A.Chattopadhyay@rug.nl

S. Plantinga
e-mail: S.Plantinga@rug.nl

G. Vegter
e-mail: G.Vegter@rug.nl

decide that the topology is correct, it subdivides the box. In other words, the certified meshing algorithm terminates after performing a sufficient number of subdivisions required to generate a certified mesh. No user-specified parameter is needed to stop the algorithm. However, interval arithmetic converges very slowly for implicit functions like (1), i.e., sums consisting of a large number of terms (translated radial basis functions). Improving the performance of the certified meshing algorithm [14], in case of RBF-interpolants, is the main goal of the current paper. Although in certified meshing 'isotopy' is the main criterion, one may also be looking for an approximating mesh which is close enough (say, with respect to the *Hausdorff distance*) to the original isosurface. To satisfy this small Hausdorff distance criterion is straightforward since for that one needs to further subdivide the boxes containing the zero set until the approximated mesh is close enough to the zero set. However, in this paper we focus on the certified meshing step.

Our contribution Although [14] presents a general scheme for certified extraction of isocurves and isosurfaces, brute force application of IA, as advocated in [14], does not yield acceptable performance if the implicit function is too complicated. This paper presents a general versatile method in case of RBF-interpolants that improves computing time drastically, yielding quadratic convergence without sacrificing the topological guarantees.

Our early experiments show that even the straightforward use of *affine arithmetic (AA)* [7], a fine tuned version of IA, does not improve running times sufficiently. Therefore, we developed several improved strategies. We use a combination of *linear* and *quadratic* upper and lower bounds for the summands in (1), exploiting the fact that each term in the sum is of the same form. This *Bounding-Plane-Bounding-Quadric (BPBQ) strategy* works for certain RBFs, and leads to a spectacular improvement of the running time, since far less subdivisions of boxes are needed before the algorithm can decide that the topology is correct. Since linear bounds are not easy to obtain for all types of RBFs, we also developed a more general method based on *quadratic* bounding functions, the *Bounding-Paraboloid (BParab) strategy*, which works for commonly used RBFs. Finally, we give pictures and performance results confirming the improved quality of the optimized strategies in terms of time and space complexity.

Related work Current methods for meshing RBF-based implicit surfaces do not come with topological guarantees, since they are usually based on the Marching Cubes algorithm [11]. Methods for certified meshes for implicit surfaces are presented in [1, 18]. In [14] interval arithmetic is used to extract certified meshing of implicit surfaces. Several improvements to standard interval arithmetic have been proposed. For an overview we refer to [12]. However, most of these improvements are restricted to algebraic functions, or do not give sufficiently accurate results to make interval arithmetic for radial basis functions practical.

In a similar context, [17] brings a new recursive Taylor method for ray-casting algebraic surfaces and shows that this method works better than various versions of interval and affine arithmetic. In general, interval or affine arithmetic cannot be used to detect degenerate zeros, like that of the function $f(x) = x^2$. For polynomials dedicated methods are available, but these are based on algebraic properties, and depart completely from the paradigm of numerical computing, cf. [4]. However, until now there is no 'certified' method in the literature to detect singularities for general implicit functions.

2 Preliminaries

2.1 Interval arithmetic (IA)

Interval arithmetic is used to cope with rounding errors in finite precision computations. A *range function* $\square F$ for a function $F : \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional input interval I (i.e., an m-box) an n-dimensional interval output $\square F(I)$, such that $F(I) \subset \square F(I)$. A range function is said to be *convergent* if the diameter of the output interval converges to 0 when the diameter of the input interval shrinks to 0. Convergent range functions exist for the basic operators and functions, so all range functions are assumed to be convergent. For an overview of interval arithmetic methods and their optimizations, we refer to [13].

2.2 Affine arithmetic (AA)

Affine arithmetic [7] is a refinement of IA based on refined tracking of the accumulating errors. In this section, we briefly discuss the range computation method using AA. In AA, each input or computed quantity x is expressed in an affine form (AF):

$$\hat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \cdots + x_n\epsilon_n,$$

where x_i are known floating point numbers and ϵ_i are symbolic variables whose values are only known to lie in the range $[-1, +1]$. Moreover, the interval $I(\hat{x})$ corresponding to the AF $\hat{x}$ is computed back as: $I(\hat{x}) = [x_0 - rad(\hat{x}), x_0 + rad(\hat{x})]$ where $rad(\hat{x}) = \sum_{i=1}^{n} |x_i|$. Thus, for example, a quantity x which is known to lie in the range $[3, 7]$ can be represented by the affine form $\hat{x} = 5 + 2\epsilon_k$, for some k. Conversely, the form $\hat{x} = 10 + 2\epsilon_3 - 5\epsilon_8$ implies that the corresponding quantity x lies in the range $[3, 17]$. The sharing of a symbol ϵ_j among two affine forms $\hat{x}, \hat{y}$ implies that the corresponding quantities x, y are partially dependent, in the sense that their joint range is smaller than

the Cartesian product of their separate ranges. For example, if $\hat{x} = 10 + 2\epsilon_3 - 6\epsilon_8$ and $\hat{y} = 20 + 3\epsilon_4 + 4\epsilon_8$, then the individual ranges of x and y are $[2, 18]$ and $[13, 27]$, but the joint range of the pair (x, y) is the hexagon with corners $(2, 27)$, $(6, 27)$, $(18, 19)$, $(18, 13)$, $(14, 13)$, $(2, 21)$, which is a proper subset of the rectangle $[2, 18] \times [13, 27]$. Moreover, affine forms can be combined with the standard arithmetic operations or elementary functions, to obtain guaranteed approximations to formulas.

Affine operations Given affine forms $\hat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \cdots + x_n\epsilon_n$, $\hat{y} = y_0 + y_1\epsilon_1 + y_2\epsilon_2 + \cdots + y_n\epsilon_n$ for x and y one can compute affine form $\hat{z} = z_0 + z_1\epsilon_1 + z_2\epsilon_2 + \cdots + z_n\epsilon_n$ corresponding to an affine operation $z = \alpha x + \beta y + \gamma$, by simply setting $z_j \leftarrow \alpha x_j + \beta y_j + \gamma$ for every j.

Non-affine operations A non-affine operation $z \leftarrow f(x, y, \ldots)$, like multiplication $z \leftarrow xy$ or $z \leftarrow \sin(x^2 + y^2)$, cannot be performed exactly, since the result would not be an affine form of the ϵ_i. In that case, one should take a suitable affine function f^a that approximates f to first order, in the ranges implied by $\hat{x}$ and $\hat{y}$; and compute $\hat{z} \leftarrow f^a(\hat{x}, \hat{y}, \ldots) + z_k\epsilon_k$, where z_k is an upper bound for the absolute error $|f - f^a|$ in that range, and ϵ_k is a new symbolic variable not occurring in any of the previous affine forms. For example, an affine approximation f^a corresponding to the function $f(x) = \sqrt{x}$ over $[c, d]$, using Chebyshev (minimax) approximation [7], is given by $f^a(x) = \alpha x + \zeta$ where $\alpha = \frac{\sqrt{d}-\sqrt{c}}{d-c}$ and $\zeta = \frac{\sqrt{c}+\sqrt{d}}{8} + \frac{\sqrt{c}\sqrt{d}}{2(\sqrt{c}+\sqrt{d})}$.

2.3 Certified meshing algorithm

Implicit functions provide a convenient way of representing smooth surfaces in 3-space. However, piecewise linear approximations are often required for computer visualization. Ordinary meshing algorithms can only compute function values at a finite number of points. Thus these schemes may miss important details of the implicit surface, and correct topology of the mesh cannot be guaranteed. The certified meshing algorithm [14] subdivides the domain of an implicit function until it approximates the zero set of the function in each box with a topologically correct piecewise linear surface. Algorithm APPROXIMATESURFACE takes an implicit function $F : \mathbb{R}^3 \to \mathbb{R}$ and a box B as input, and computes a piecewise linear approximation of $F^{-1}(0) \cap B$, assuming that the zero set $F^{-1}(0)$ of F contains no singular points of F inside B. It uses range functions for F and its gradient ∇F over an interval to extract information about the surroundings of the grid points.

Algorithm 2.1: APPROXIMATESURFACE(F, B)

1. Initialize octree T to B;
2. Subdivide T until for all leaves I :
   $0 \notin \Box F(I) \vee \langle \Box\nabla F(I), \Box\nabla F(I)\rangle > 0$;
3. BALANCEOCTREE(T);
4. MESH(T).

Here, MESH(T) approximates the zero level set inside the box T by a piecewise linear function. The first clause in line 2 discards cells I for which $0 \notin \Box F(I)$, i.e., boxes which are guaranteed not to contain part of the zero set of F. The second clause implies that $\langle \nabla F(p), \nabla F(q)\rangle > 0$, for all $p, q \in I$, so the direction of the gradient (and, therefore, of the normal of the curve) does not change by more than $\pi/2$ over this box. This implies that the zero set of F is *parametrizable* (i.e., can be written as the graph of a function of x, y, or y, z, or x, z), which is the key property in the proof of topological correctness of the output. To facilitate the meshing of the resulting octree, it is first balanced (line 3). Balancing means extra subdivision of the octree leaves until neighboring leaves differ at most one level in depth. MESH (line 4) generates a triangulated surface using a tetrahedral subdivision of the ambient space. For details we refer to [14]. We also have a 2D-version of this algorithm, which computes topologically correct polygonal approximations of regular implicit curves.

3 Range functions for RBFs

The range intervals computed using IA for RBF-interpolants s (of the form in (1)) are too conservative. Therefore the algorithm APPROXIMATECURVE needs a large number of subdivisions (of the domain interval) before it satisfies the stopping criteria of the algorithm. Thus, for RBF-based implicit functions an IA-based implementation of algorithm APPROXIMATESURFACE(s, I) has unacceptable running time. Our goal is to improve the performance considerably by optimizing the range intervals $\Box s(I)$ and $\Box\nabla s(I)$ for such RBF-interpolants s on a box I.

In Sect. 3.1 we present the *Bounding-Paraboloid* (*BParab*) *strategy* for finding range intervals of RBF-interpolants. This strategy works for almost all commonly used RBFs and in any dimension. More precisely, this strategy works with all C^2-smooth basis functions. For the thin plate spline $\varphi = r^2 \log r$, being only C^1, the method works only for computing $\Box s(I)$ but not for computing $\Box s_x(I)$ or $\Box s_y(I)$, and, hence, fails for the strategy BParab.

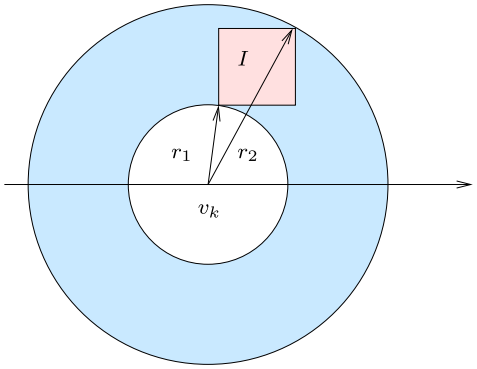


Fig. 1 Near and far point of a square interval I. If the center v_k lies inside the box I, then $r_1 = 0$

Subsequently, in Sect. 3.2 we discuss the *Bounding-Plane-Bounding-Quadric (BPBQ) strategy* for the cubic RBF, which is widely used in reconstruction of geometric surfaces from scattered point samples [3, 8, 9, 16, 20]. It turns out that in many experiments this BPBQ-strategy for the cubic RBF works better in practice than the BParab-strategy.

Finally, in Sect. 3.4, we present a greedy strategy which finds better range intervals by first subdividing the interval into small intervals. Combined with the other strategies this results in better timing for isotopic meshing.

3.1 The Bounding-Paraboloid (BParab) strategy

Computing $\square s(I)$ Our first optimization strategy determines a lower bound $l_k(\underline{x})$ and an upper bound $u_k(\underline{x})$ for $w_k\varphi(\|\underline{x} - v_k\|)$ on the box I, such that the minimal value $L(I)$ of $l(\underline{x}) := \sum_k l_k(\underline{x}) + p(\underline{x})$ on I and the maximum value $U(I)$ of $u(\underline{x}) := \sum_k u_k(\underline{x}) + p(\underline{x})$ on I are easy to compute. Moreover, l and u are chosen in such a way that $\square s(I) = [L(I), U(I)]$ yields a much better range interval than IA, or even AA.

Our approach is based on the observation that the summand $w_k\varphi(\|\underline{x} - v_k\|)$ is radially symmetric with respect to the center v_k. We will find *quadratic* upper and lower bounds for the *univariate* function $w_k\varphi(r)$, for r ranging over the smallest interval $J_k = [r_1, r_2]$ for which $r_1^2 \le \|\underline{x} - v_k\|^2 \le r_2^2$, for all $\underline{x} \in I$. See Fig. 1. More precisely, the univariate upper bound of $w_k\varphi(r)$ on J_k is of the form $\alpha_k r^2 + \beta_k$, yielding $s(\underline{x}) \le \sum_{k=1}^n \alpha_k \|\underline{x} - v_k\|^2 + \sum_{k=1}^n \beta_k + p(\underline{x})$, for $\underline{x} \in I$. Since, for most RBFs, the polynomial p has degree at most two, the upper bound u is a bivariate or trivariate quadratic function, obtained by adding the coefficients of the upper bounds for each individual summand. Moreover, a straightforward way to compute a conservative upper bound $U(I)$ of u on I is by using interval arithmetic as $U(I) = \text{UPPER}(u(I))$ (cf. [2]). A quadratic lower bound l for the RBF-interpolant s on I is determined similarly. Again, a conservative lower bound $L(I)$ of l on I is computed using interval arithmetic as $L(I) = \text{LOWER}(l(I))$ (cf. [2]). Here we note that one can compute more accurate bounds $U(I)$ and $L(I)$ by computing a global maximum of u over I or a global minimum of l over I. In view of the special shape of the quadratic upper and lower bounds this approach is called the *bounding paraboloid (BParab) strategy*.

To determine the coefficients α_k and β_k we first determine a quadratic function of the form $r \mapsto \alpha_k r^2 + \gamma_k$, which interpolates the radial basis function at the end-points of the interval J_k. In other words, we take

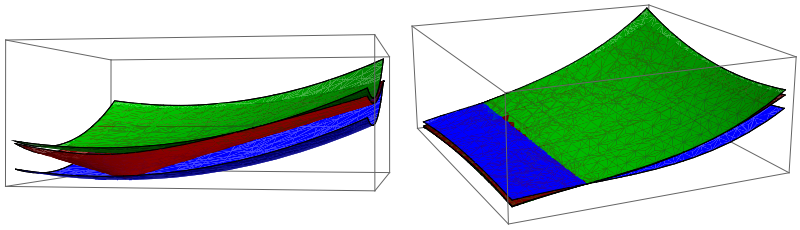
$$\alpha_k = \frac{w_k(\varphi(r_2) - \varphi(r_1))}{r_2^2 - r_1^2}.$$

Algorithm 3.1: BPARAB-BOX-S(I)

1. Initialize: $u(\underline{x}) \leftarrow p(\underline{x})$, $l(\underline{x}) \leftarrow p(\underline{x})$;
2. **for** $k \leftarrow 1$ **to** n
   do
   - 2.1. Compute $J_k \leftarrow [r_{k1}, r_{k2}]$ with $r_{k1}^2 \le \|\underline{x} - v_k\| \le r_{k2}^2$ over I;
   - 2.2. $\alpha_k \leftarrow \frac{w_k(\varphi(r_{k2}) - \varphi(r_{k1}))}{r_{k2}^2 - r_{k1}^2}$;
   - 2.3. $\beta_k \leftarrow \max_{J_k}(w_k\varphi(r) - \alpha r^2)$;
   - 2.4. $\beta_k' \leftarrow \min_{J_k}(w_k\varphi(r) - \alpha r^2)$;
   - 2.5. $u(\underline{x}) \leftarrow u(\underline{x}) + \alpha_k\|\underline{x} - v_k\|^2 + \beta_k$;
   - 2.6. $l(\underline{x}) \leftarrow l(\underline{x}) + \alpha_k\|\underline{x} - v_k\|^2 + \beta_k'$;
3. $U(I) \leftarrow \text{UPPER}(u(I))$;
4. $L(I) \leftarrow \text{LOWER}(l(I))$;
5. **return** $([L(I), U(I)])$.

An upper bounding quadratic function of the form $r \mapsto \alpha_k r^2 + \beta_k$ is now obtained by taking $\beta_k \ge \max_{r \in J_k}(w_k\varphi(r) - \alpha r^2)$. We note that the expression $w_k\varphi(r) - \alpha r^2$ is a univariate continuous function over J_k. Thus β_k is computed analytically as the global maximum of $w_k\varphi(r) - \alpha r^2$ over J_k, for any particular basis function φ. Similarly, a lower bounding quadratic function of the form $r \mapsto \alpha_k r^2 + \beta_k'$ is found by taking $\beta_k' \le \min_{r \in J_k}(w_k\varphi(r) - \alpha r^2)$. A general approach for finding optimal α and β is presented in Appendix. Algorithm 3.1 gives the corresponding pseudocode for computing $\square s(I)$.

Computing $\square \nabla s(I)$ We focus on the 2D-case for simplicity, but the 3D-case is similar. To find optimal ranges $\square s_x(I)$



(a) (b)

Fig. 2 (**a**) Bounding paraboloids corresponding to graph of ψ. (**b**) Graphs of cubic polynomials U_{k0} (*green*) and L_{k0} (*blue*) corresponding to Φ_{kx} (*red*) when $v_{kx} \in (x_1, x_2)$ over $I \equiv [x_1, x_2] \times [y_1, y_2]$

and $\square s_y(I)$ (and $\square s_z(I)$ in the trivariate case) for the components of the gradient of the RBF-interpolant (1), first note that s_x is given by

$$s_x(\underline{x}) = \sum_{k=1}^{n} w_k \frac{\varphi'(\|\underline{x} - v_k\|)}{\|\underline{x} - v_k\|} (x - v_{kx}) + p_x(\underline{x}), \tag{2}$$

where $\underline{x} = (x, y)$ and $v_k = (v_{kx}, v_{ky})$ (or $\underline{x} = (x, y, z)$ and $v_k = (v_{kx}, v_{ky}, v_{kz})$ in the trivariate case). First we consider the case $v_{kx} \notin (x_1, x_2)$, where $I \equiv [x_1, x_2] \times [y_1, y_2]$. Applying the same approximation strategy as before to find quadratic lower bounds on the one-dimensional interval J_k for each of the *univariate factors* $w_k \varphi'(r)/r$, leads to a bivariate (or trivariate) *cubic* lower bound $L_k(\underline{x})$ on the box I for the kth summand in (2) of the form: $L_k(\underline{x}) = a_k\, x\, (x^2 + y^2) + Q_k(\underline{x})$, where a_k is a real constant and $Q_k(\underline{x})$ is a quadratic polynomial. A cubic upper bound $U_k(\underline{x})$ of this form is found similarly.

We now focus on the case $v_{kx} \in (x_1, x_2)$, where $I \equiv [x_1, x_2] \times [y_1, y_2]$. Figure 2 shows the graph of kth term

$$\Phi_{kx}(x, y) = w_k \frac{\varphi'(\|\underline{x} - v_k\|)}{\|\underline{x} - v_k\|} (x - v_{kx})$$

of s_x and the graphs of the corresponding bivariate cubic polynomials, say L_{k0} and U_{k0}, which are obtained by multiplying the linear term $x - v_{kx}$ with the lower and upper bounds of

$$w_k \frac{\varphi'(\|\underline{x} - v_k\|)}{\|\underline{x} - v_k\|}$$

over I, respectively. Clearly, in this case L_{k0} and U_{k0} are not lower or upper bounds of Φ_{kx} over I. To find the lower and upper bounds we proceed as follows. First, we find an average of polynomials L_{k0} and U_{k0}, which is again a bivariate cubic polynomial,

$$Q_{\text{avg}}(\underline{x}) = \frac{1}{2}\big(U_{k0}(\underline{x}) + L_{k0}(\underline{x})\big).$$

Q_{avg} is a good approximation of $\Phi_k(x, y)$ over I. Then we translate the graph of Q_{avg} downwards (in the z-direction) so that it lies below the graph of L_{k0}, to obtain a lower bounding cubic polynomial L_k of $\Phi_{kx}(x, y)$ over I. Similarly, the graph of Q_{avg} is translated upwards (in the z-direction) so that it lies above the graph of U_{k0}, to obtain an upper bounding cubic polynomial U_k of $\Phi_{kx}(x, y)$ over I.

A straightforward use of interval arithmetic (as in the case of $\square s$) yields the minimal value of $\sum_k L_k(\underline{x}) + p_x(\underline{x})$ and the maximal value of $\sum_k U_k(\underline{x}) + p_x(\underline{x})$ on I, and, hence, a good interval $\square s_x(I)$. A good interval $\square s_y(I)$ is computed similarly. As we will show in Sect. 4, this strategy improves the performance of the certified meshing algorithm ApproximateSurface considerably for various RBFs. The pseudocodes corresponding to $\square s_x(I)$ and $\square s_y(I)$ are similar as in BParab-Box-s.

Remark 1 We can increase the computational accuracy using the BParab strategy by generalizing the approximating radially symmetric polynomial $r \mapsto \alpha_k r^2 + \beta_k$ by $r \mapsto \alpha_{k0} r^{2n} + \alpha_{k1} r^{2n-2} + \cdots + \alpha_{kn}$. In other words, we need to interpolate the functions $\phi(r)$ or $\psi(r)$ at $n+1$ distinct points of the interval J. Depending on the choice of RBFs one can choose the distribution of the interpolating points over the interval J [15]. Subsequently, we get multivariate bounding polynomials of degree $2n$ for the RBF-interpolants and multivariate bounding polynomials of degree $2n + 1$ for the partial derivatives of the RBF-interpolants. However, it is clear that using higher degree polynomial is computationally more expensive. Therefore, there is a trade-off between computational accuracy and speed of the algorithm. We do not pursue this issue further in this paper.

3.2 Bounding-Plane-Bounding-Quadric (BPBQ) strategy for the cubic RBF

The cubic RBF, given by $\varphi(r) = r^3$, is used widely in reconstruction of geometric surfaces from scattered point samples. Therefore, for this case we designed an even better strategy based on special properties of this RBF, like convexity. More precisely, in this case we find *linear* upper and lower bounds for the RBF-interpolant, which are good linear approximations for the function on the box. Finding linear approximations for the partial derivatives is much harder, so we focus on obtaining good *quadratic* upper and lower bounds. Therefore, this approach is called the *Bounding-Plane-Bounding-Quadric (BPBQ) strategy*.

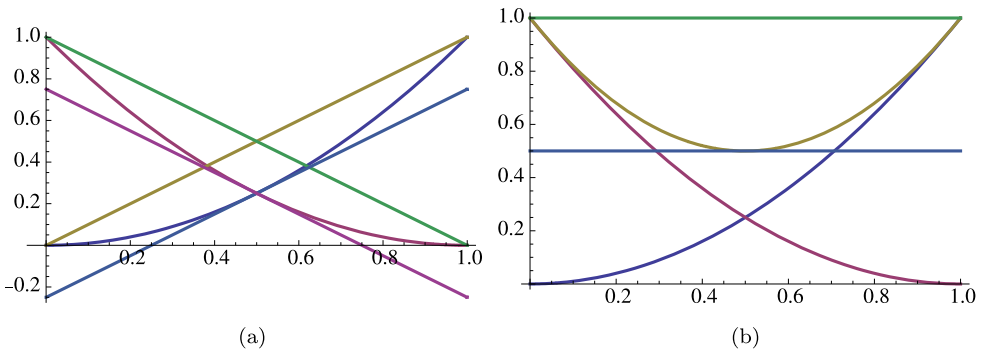


Fig. 3 Linear upper and lower bounds of the (**a**) functions $f(x) = x^2$ and $g(x) = (1-x)^2$, (**b**) sum function $h \equiv f + g$

To illustrate that bounding planes can give better range intervals than AA, which in turn works better in practice than IA, we consider the following simple example. Consider the problem of finding the range interval of the sum $h(x)$ of $f(x) = x^2$ and $g(x) = (1-x)^2$ over $I = [0, 1]$. Using IA, the sum is $\square f(I) + \square g(I) = [0, 1] + [0, 1] = [0, 2]$, whereas using AA this interval is computed as follows. First, the unknown quantity x is expressed as affine form (AF) $\hat{x} = 0.5 + 0.5\epsilon_1$. Then, AFs corresponding to $f(x)$ and $g(x)$ are respectively, $\hat{z}_1 = 0.25 + 0.5\epsilon_1 + 0.25\epsilon_2$ and $\hat{z}_2 = 0.25 - 0.5\epsilon_1 + 0.25\epsilon_3$. Therefore, the AF corresponding to $h(x)$ is $\hat{z} = 0.5 + 0\epsilon_1 + 0.25\epsilon_2 + 0.25\epsilon_3$. Hence, the range of $h(x)$ over $[0, 1]$ is $[0, 1]$.

Now using the bounding plane strategy, the linear upper bounds of $f(x)$ and $g(x)$ over I are $u_1(x) = x$ and $u_2(x) = 1 - x$, respectively. Similarly, linear lower bounds are $l_1(x) = x - 0.25$ and $l_2(x) = -x + 0.75$, respectively. Hence, the upper and lower bounds of the sum $h(x)$ over I are $u(x) = 1$ and $l(x) = 0.50$, respectively (Fig. 3). Therefore, the range interval of h over I is $[0.50, 1]$, which is better than the range interval using AA.

Computing $\square s(I)$ First, we observe that using the convexity property of the cubic RBF it is possible to group the summands of the RBF-interpolant s, given by (1), into a convex function s^+ and a concave function s^-:

$$s = s^+ + s^-. \tag{3}$$

Here s^+ corresponds to the sum of the terms with positive weights Here s^+ corresponds to the sum of the terms with positive weights and the linear term, and s^- corresponds to the sum of the terms with negative weights. Grouping reduces the number of bounding plane computations from $O(n)$ to a constant number, where n is the number of terms in the RBF-interpolant. Next, we find linear lower and upper bounds for each of the grouped functions in (3) over I, by straightforward geometric computation. More specifically, a lower bound of the function s^+ over I is obtained by computing the tangent of its graph at the middle point of I. Again, the upper bound is taken as the plane passing through one of the diagonals (obtained by joining the corner points of the graph of s^+ over I) and parallel to the other diagonal so that the plane lies above the graph of s^+. Thus if $l_\pm$ and $u_\pm$ are the lower and upper bounding linear functions for $s^\pm$ over I, then $l_- + l_+$ and $u_- + u_+$ are lower and upper bounds of s over I. Computing the minimum of this lower bound and the maximum of the upper bound over the box I we obtain a good range interval $\square s(I)$. Algorithm 3.2 gives the pseudocode for computing $\square s(I)$.

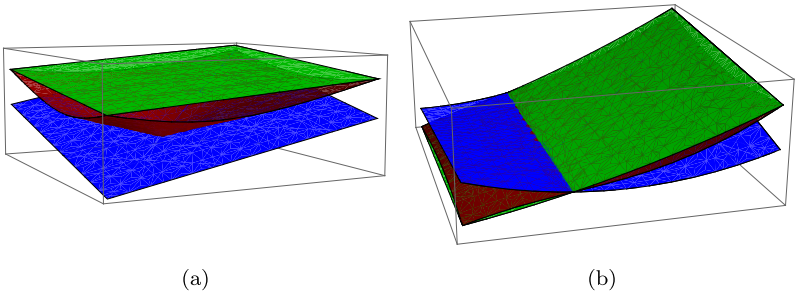
Algorithm 3.2: BPBQ-BOX-S(I)

1. $u^+ \leftarrow$ UPPERBOUND(s^+);
2. $l^+ \leftarrow$ LOWERBOUND(s^+);
3. $u^- \leftarrow$ UPPERBOUND(s^-);
4. $l^- \leftarrow$ LOWERBOUND(s^-);
5. $u \leftarrow u^+ + u^-$;
6. $l \leftarrow l^+ + l^-$;
7. $U(I) \leftarrow$ UPPER($u(I)$);
8. $L(I) \leftarrow$ LOWER($l(I)$);
9. **return** ($[L(I), U(I)]$).

Computing $\square \nabla s(I)$ To find optimal ranges $\square s_x(I)$ and $\square s_y(I)$ for the components of the gradient of the RBF-interpolant (1), first note that the partial derivative s_x of the cubic RBF is given by

$$s_x(\underline{x}) = \sum_{k=1}^{n} 3w_k \|\underline{x} - v_k\|(x - v_{kx}) + p_x(\underline{x}), \tag{4}$$

where $\underline{x} = (x, y)$ and $v_k = (v_{kx}, v_{ky})$. Here, each summand $3\|\underline{x} - v_k\|(x - v_{kx})$ is the product of a distance function $\|\underline{x} - v_k\|$ and a linear function. Finding a linear upper and lower bound of the distance function is straightforward, and leads to a bivariate quadratic lower bound of each summand



(a) (b)

Fig. 4 (**a**) Bounding planes of the graph of distance function. (**b**) Graphs of quadratic polynomials U_{k0} (*green*) and L_{k0} (*blue*) corresponding to Φ_{kx} (*red*) when $v_{kx} \in (x_1, x_2)$ over $I \equiv [x_1, x_2] \times [y_1, y_2]$

of the form $L_k(\underline{x}) = x\, h_k(\underline{x}) + l_k(\underline{x})$, where h_k and l_k are linear functions. A quadratic upper bound $U_k(\underline{x})$ of this form is found similarly.

Similarly as in the BParab strategy, we discuss the case when $v_{kx} \in (x_1, x_2)$, where $I \equiv [x_1, x_2] \times [y_1, y_2]$. Figure 4 shows the graph of the kth term $\Phi_{kx}(x, y) = \|\underline{x} - v_k\|(x - v_{kx})$ of s_x and the corresponding bivariate quadratic polynomials L_{k0} and U_{k0} over I. Clearly, in this case L_{k0} and U_{k0} are not the lower or upper bounds of Φ_{kx} over I. Therefore, to find the lower and upper bounds we use the same technique as in the BParab strategy.

A straightforward use of interval arithmetic (as in the previous cases) yields the minimal value of $\sum_k L_k(\underline{x}) + p_x(\underline{x})$ and the maximal value of $\sum_k U_k(\underline{x}) + p_x(\underline{x})$ on I, and, hence, a good interval $\square s_x(I)$. A good interval $\square s_y(I)$ is computed similarly. The corresponding pseudocodes are similar as in the BParab strategy (Algorithm 3.1). Experiments with the BPBQ-strategy corroborate this improvement; see Sect. 4.

3.3 Convergence

Convergence of the certified meshing algorithm ApproximateSurface(F, B) depends on the accuracy of the computed range intervals. Therefore, to compare the performance of the algorithm, using different range computation strategies, one needs to compare the accuracy of the range intervals in different strategies. Usually, computed range intervals are wider than actual (ideal) range intervals. Now the error in the computed range interval is determined by: (1) the approximation error, and (2) the round-off error, respectively. The approximation error, which is caused by approximation of the given function with some other suitable function, is dominant in the error term if the input interval size is sufficiently large. However, the approximation error goes down when the input interval size becomes smaller. Although if the size of the input interval becomes arbitrarily small the round-off error dominates in the error term. In the certified meshing algorithm, our aim is to reduce the approximation error, so that the subdivision process terminates while the input intervals are wide enough and we can ignore the round-off error.

Interval Arithmetic yields approximation errors that are linear in the size of the input interval, whereas the approximation errors for Affine Arithmetic are quadratic [19] in the size of the input interval. Therefore, as the size of the input intervals gets smaller, the approximation error in AA becomes less important than IA. In other words, when the input intervals become smaller the computed range intervals in AA converge faster (than in IA) to the actual range interval. Next we find dependency of the approximation error in the computed range intervals, using BPBQ and BParab strategy, on the size $\|I\|$ of the input intervals I defined as $\|I\| := \max_{x,y \in I} \|x - y\|$. So, $\|I\|$ is the diameter of I.

Lemma 1 *If the RBF φ is C^2, then the approximation error in the range interval using the BPBQ strategy depends quadratically on the size of the input interval.*

Proof The linear bound of $\varphi(\underline{x})$ over I is of the form $l(\underline{x}) = \varphi(\underline{x}_0) + \nabla\varphi(\underline{x}_0)(\underline{x} - \underline{x}_0)$ where $\underline{x}_0$ is the center of the interval I. Since $\varphi(\underline{x}) - l(\underline{x}) = O(\|\underline{x} - \underline{x}_0\|^2)$, we see that $\varphi(\underline{x}) - l(\underline{x}) = O(\|I\|^2)$. □

Next, we determine the approximation error in the BParab strategy.

Lemma 2 *If the RBF φ is C^2, the approximation error in the range interval using the BParab strategy with any radial basis function depends quadratically on the size of the input interval.*

Proof First we find the approximation error in the interpolation of the one-variable radially symmetric function $\varphi(r)$ with a radially symmetric quadratic function $q(r) = \alpha r^2 + \beta$ over $J := [a, b]$ (which corresponds to a higher dimensional interval I, as described before). Now, since $q(a) = \varphi(a)$ and $q(b) = \varphi(b)$, the error term of the interpolation is given by

$$e(r) := \varphi(r) - q(r) = \frac{1}{2!}(r - a)(r - b)\varphi^{(2)}(\xi)$$

for some $\xi \in J$. Therefore, when $\|J\| \to 0$, which is implied by $\|I\| \to 0$, the error satisfies

$$\varphi\big(\|\underline{x}\|\big) - q\big(\|\underline{x}\|\big) = O\big(\|I\|^2\big) \text{ for } \underline{x} \in I.$$

Hence, the error in the range interval using the BParab strategy depends quadratically on the size of the input interval. □

Remark 2 It is worthwhile to mention that both AA and the optimized methods (BParab and BPBQ) have quadratic convergence. Although, since BParab and BPBQ compute range intervals for RBF-interpolants more accurately compared to AA, the subdivision time in the algorithm APPROXIMATESURFACE is considerably less for BParab and BPBQ (see the experimental results in Sect. 4).

Remark 3 We note that using the generalized version of BParab strategy we can have convergence of any desired order. Suppose we approximate an RBF $\varphi \in \mathcal{C}^{n+1}$ by a radially symmetric polynomial of degree $2n$, i.e., of the form $r \mapsto \alpha_{k0} r^{2n} + \alpha_{k1} r^{2n-2} + \cdots + \alpha_{kn}$, where α_{ki} are unknown real numbers to be found by some interpolation scheme. By a similar error analysis as before one finds that in this case the error term of the approximation is $O(\|I\|^{n+1})$.

3.4 Subdivision strategy

To improve the performance even further, we also experimented with a hybrid approach in which the BParab and BPBQ strategies were extended by a preliminary subdivision of the boxes. Since $\cup_{i=1}^{m} \square s(I_i) \subseteq \square s(\cup_{i=1}^{m} I_i)$, the range intervals $\square s(I)$ and $\square \nabla s(I)$ might become smaller by first subdividing the interval I into m subintervals $I_1, \ldots, I_m$, followed by computing the range intervals $\square s(I_i)$ for each subinterval I_i. The number of subintervals, m, is called the *subdivision number (SN)*.

To illustrate that the subdivision strategy can give better range intervals, we consider the example of finding the range interval of the function $f(x) := x^2$ over $I \equiv [-1, 2]$. Using IA, we find the range interval $[-2, 4]$. Now let us subdivide domain interval I into $I_1 = [-1, 0]$, $I_2 = [0, 1]$ and $I_3 = [1, 2]$. Applying IA on each of the intervals I_1, I_2 and I_3, we find the range intervals $[0, 1]$, $[0, 1]$ and $[1, 4]$, respectively. Finally, the union of these three range intervals is $[0, 4]$, which is better than the range interval $[-2, 4]$ obtained using IA.

In general, preliminary subdivision reduces the size of the range intervals $\square s(I_i)$, resulting in a smaller number of leaves of the octree. On the other hand, if the boxes get smaller, the approximation error goes down, but the round-off error becomes significant. Therefore, the accuracy does not improve significantly, whereas the computing time does increase. So we may expect the computing time to decrease up to a *critical depth (CR)* of the subdivision process. Beyond this point further subdivision is expected to lead to increased computing times. In Sect. 4 we show experimental evidence for this observation.

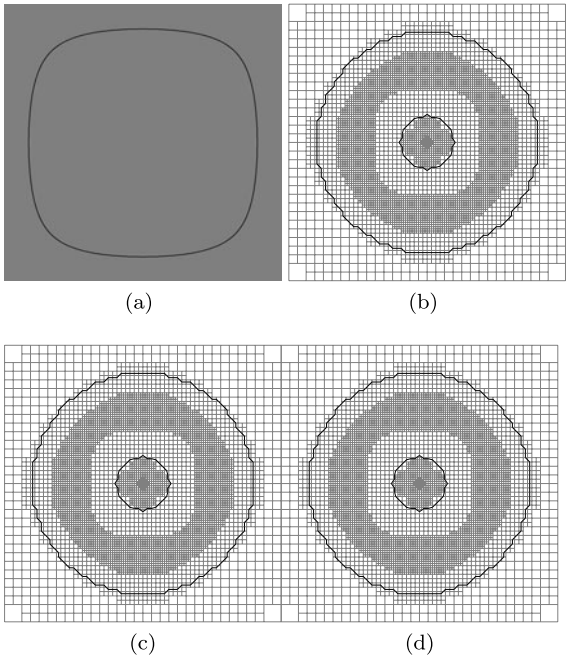


Fig. 5 Isocurve extraction for a cubic RBF-interpolant (100 centers) of the function $f(x, y) = xy(x-1)(y-1) - 0.02$, sampled uniformly on the square $[0, 1] \times [0, 1]$ using strategies: (**a**) IA, (**b**) AA, (**c**) BPBQ and (**d**) BParab

4 Experimental results

In the previous section we have designed different optimization strategies for finding the range intervals of RBF-interpolants. In this section we compare the efficiency and performance of certified meshing algorithm implemented using those range intervals. We experiment with both two- and three-dimensional RBF-interpolants. Here we note that our main criterion is topological correctness of the approximated curves or surfaces, but that arbitrary numerical accuracy can be obtained by further subdivision of the relevant boxes if necessary.

The 2D case We present some 2D experiments using the 2D version of algorithm APPROXIMATESURFACE, implementing range functions based on IA, AA, BParab and, for the cubic RBF, the BPBQ-strategy. We extract the certified zero sets of various RBF-interpolants, and compare the number of leaves (NOL) of the subdivision tree as well as the CPU-time (CPU). The RBF-interpolants are constructed using uniform sample interpolation points extracted from several functions, over a bounding box, cf. [3, 10, 20].

Experiments with the cubic RBF Our first sequence of experiments has been performed using cubic-based interpolants. In other words, we used the RBF given by $\varphi(r) = r^3$. Tables 1, 2, 3 present the measured performance for different optimization strategies. Figures 5, 6, 7, 8, 9,

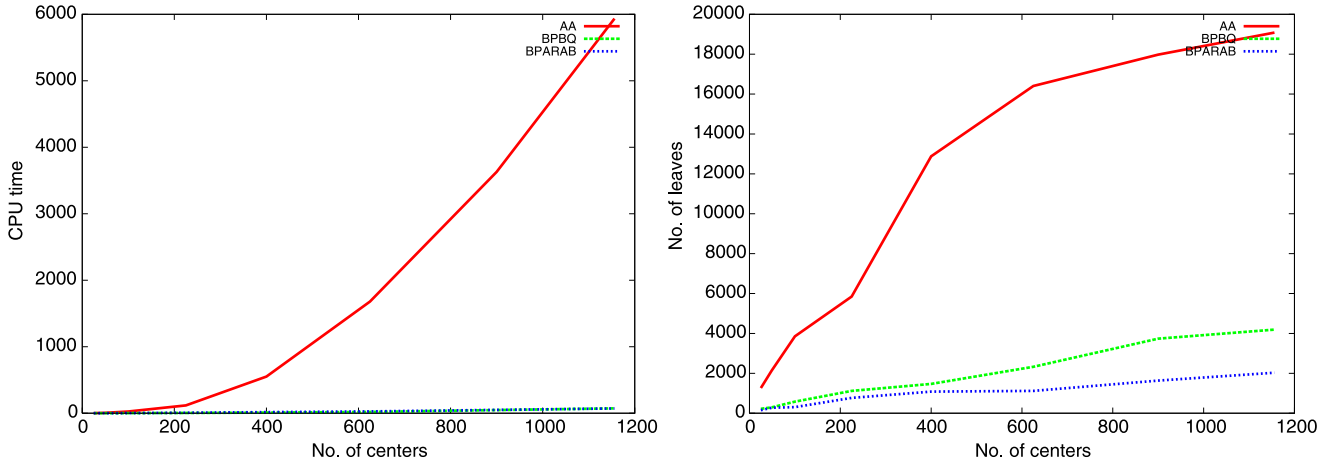


Fig. 6 Graphical representation of Table 1: (i) number of centers vs. CPU-time, (ii) number of centers vs. number of leaves of the subdivision tree

Table 1 Space and time complexity corresponding to Fig. 5

NOC	IA		AA		BPBQ		BParab	
	NOL	CPU	NOL	CPU	NOL	CPU	NOL	CPU
25	138616	6.54 s	1264	1.6 s	220	0.10 s	154	0.15 s
49	484660	39.6 s	2140	5.3 s	292	0.32 s	274	0.49 s
100	1726852	4 m 13 s	3856	25.8 s	580	1.02 s	304	1.19 s
225	6757600	36 m 47 s	5848	1 m 56 s	1120	4.46 s	769	6.30 s
400	–	–	12880	9 m 11 s	1468	9.50 s	1081	16.5 s
625	–	–	16408	27 m 59 s	2329	19.36 s	1120	27.6 s
900	–	–	17980	60 m 29 s	3736	47.76 s	1636	49.5 s
1156	–	–	19084	98 m 55 s	4192	1 m 13 s	2032	1 m 11 s

Table 2 Space and time complexity corresponding to Fig. 7

NOC	AA		BPBQ		BParab	
	NOL	CPU	NOL	CPU	NOL	CPU
25	2908	4.07 s	448	0.21 s	448	0.39 s
49	6820	16.5 s	1036	0.82 s	580	1.10 s
100	9052	55.7 s	1276	1.72 s	1156	3.70 s
225	18808	5 m 36 s	2884	9.39 s	1528	10.5 s
400	24988	17 m 16 s	3688	17.22 s	1972	29.0 s
625	31492	46 m 57 s	4264	30.43 s	3652	1 m 10 s
900	34888	108 m	5188	49.70 s	4120	1 m 46 s
1156	37288	198 m	6016	1 m 22 s	4540	2 m 40 s

10 contain the corresponding isocurves, together with the boxes corresponding to the leaf-nodes of our subdivision tree; and the respective graphical representations of the Tables 1–3. Note that Table 1 shows that straightforward use of IA does not lead to convergence (in reasonable time), except in trivial cases. Therefore, we discard IA from our remaining experiments. Moreover, from Fig. 5(c) and (d) we observe that the algorithm needs extra subdivision in some parts of the domain of the RBF-interpolant where there is no zero set of the RBF-interpolant. This extra subdivision is required for satisfying the second predicate (i.e. the small normal variation condition) in the certified meshing algorithm ApproximateSurface (or ApproximateCurve for the 2D-case).

Table 3 Space and time complexity corresponding to Fig. 9

NOC	AA		BPBQ		BParab	
	NOL	CPU	NOL	CPU	NOL	CPU
25	964	1.25 s	148	0.10 s	142	0.14 s
49	1852	5.43 s	301	0.36 s	286	0.56 s
100	3700	25.23 s	640	1.22 s	526	2.06 s
225	7288	2 m 35 s	1156	4.95 s	1246	11.82 s
400	12556	10 m 32 s	1888	13.90 s	1450	22.88 s
625	19708	35 m 19 s	2809	31.17 s	2182	51.64 s
900	25936	89 m	3940	58.34 s	3358	1 m 54 s
1156	28666	155 m	4714	1 m 24 s	4048	2 m 42 s

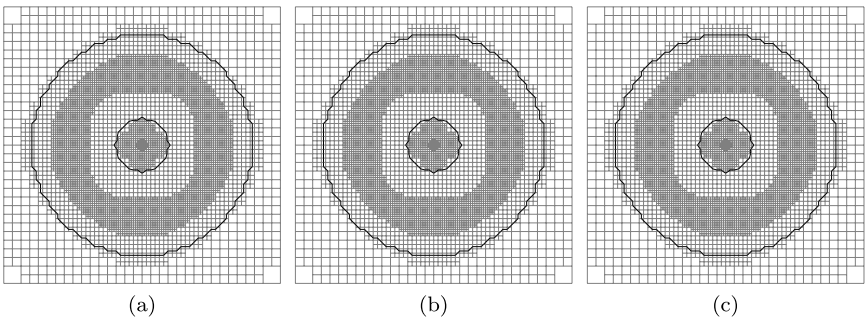


Fig. 7 Isocurve extraction for a cubic RBF-interpolant (100 centers) of the function $f(x, y) = (x^2+y^2)(1-\sqrt{x^2+y^2})-0.04$, sampled uniformly on the square $[-1.2, 1.2] \times [-1.2, 1.2]$ using strategies: (**a**) AA, (**b**) BPBQ and (**c**) BParab

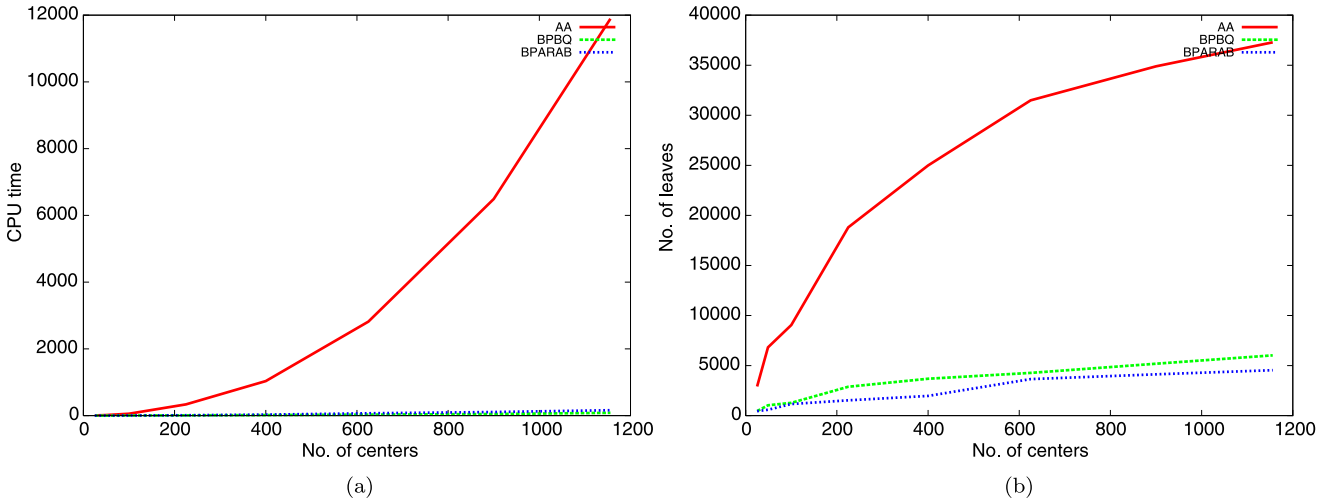


Fig. 8 Graphical representation of Table 2: (**a**) number of centers vs. CPU-time, (**b**) number of centers vs. number of leaves of the subdivision tree

We used the `Boost` interval arithmetic library [2] for IA, and the affine arithmetic library [5] for AA. All experiments have been performed on a 3-GHz Intel Pentium 4 machine under Linux with 1-GB RAM using the `g++` compiler, version 3.3.5.

Experiments with the multiquadric RBF Next, we present some more experiments using the multiquadric RBF given by $\varphi(r) = \sqrt{1+r^2}$. Tables 4 and 5 compare the performance of the AA and BParab strategies for this case. Figures 11 and 12 contain the corresponding isocurves. From these experiments it is clear that, in practice, BParab strategy works much better than AA.

Experiments using the BPBQ-strategy with subdivision Figure 13 and the corresponding Table 6 compare experimental results with the subdivision strategy, cf. Sect. 3.4, for different subdivision numbers (SN). Figure 13 shows that

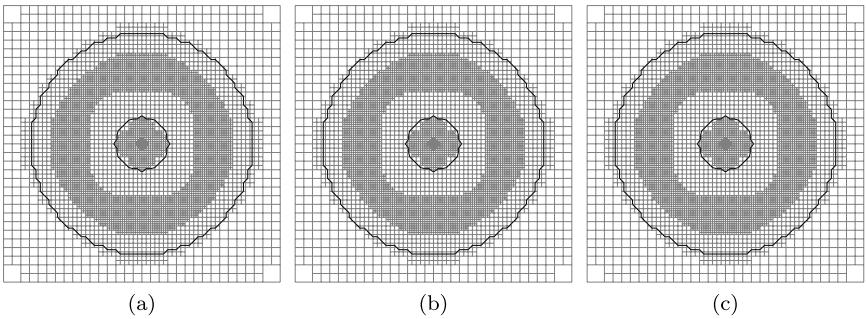


Fig. 9 Isocurve extraction for a cubic based RBF-interpolant (100 centers) of the function $f(x, y) = 4y^2 - (x+1)^3(1-x)$, sampled uniformly on the square $[-1.1, 1.1] \times [-1.1, 1.1]$ using strategies: (**a**) AA, (**b**) BPBQ and (**c**) BParab

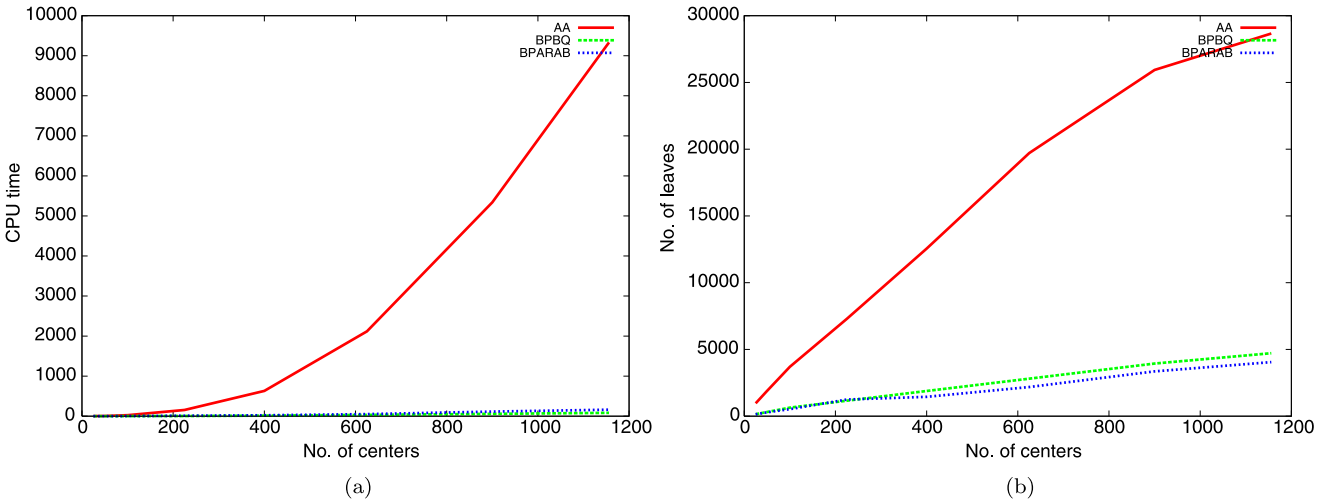


Fig. 10 Graphical representation of Table 3: (**a**) number of centers vs. CPU-time, (**b**) number of centers vs. number of leaves of the subdivision tree

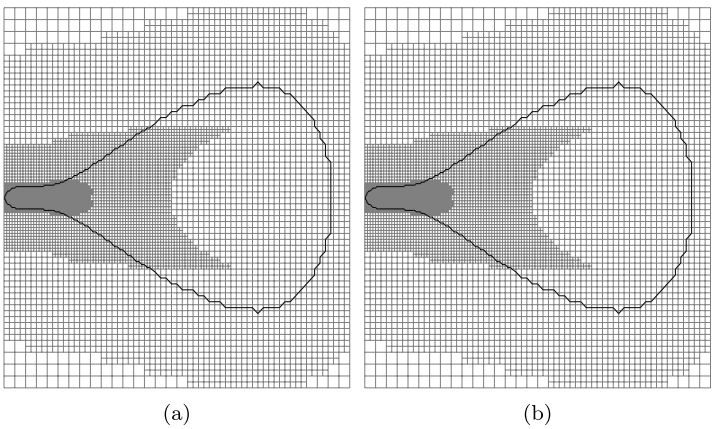


Fig. 11 Isocurve extraction for a multiquadric RBF-interpolant (49 centers) of the function $4y^2 - (x+1)^3(1-x)$, sampled uniformly on the square $[-1.1, 1.1] \times [-1.1, 1.1]$ using strategies: (**a**) AA and (**b**) BParab

the meshing algorithm needs a small number of leaves as *SN* increases since the range intervals become more accurate after subdivision. Figure 14 gives a graphical representation of Table 6. Figure 14(b) shows that the number of leaves decreases with the increase of *SN*, whereas Fig. 14(a) shows that with the increase of *SN* initially the total CPU-time of the meshing algorithm decreases, although, after some critical *SN* (here, $SN = 4^2$), the CPU-time increases again. These experiments illustrate a trade-off between the computational accuracy and the total time of convergence. Note that in all these experiments the critical depth (CR) has been chosen as the height of the subdivision tree.

The 3D case We present some experimental results of 3D implicit surface meshing with algorithm ApproximateSurface. We use the most widely used cubic RBF for constructing the implicit functions. The range intervals are

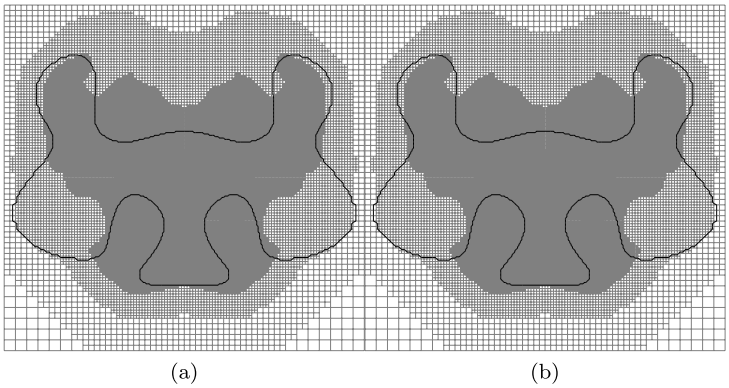


Fig. 12 Isocurve extraction for a multiquadric RBF-interpolant (25 centers) of the function $f(x, y) = (y - x^2 + 1)^4 + (x^2 + y^2)^4 - 1 = 0$, sampled uniformly on the square $[-1.25, 1.25] \times [-1.5, 1.0]$ using (**a**) AA and (**b**) BParab

Table 4 Complexity of multiquadric-based meshing corresponding to Fig. 11

NOC	AA		BParab	
	NOL	CPU	NOL	CPU-time
25	1216	1.48 s	52	0.07s
49	7726	20.8 s	154	0.30 s
100	115312	13 m 36 s	274	1.12 s
225	–	–	289	2.71 s
400	–	–	520	8.66 s
625	–	–	598	15.8 s
900	–	–	610	22.7 s
1156	–	–	874	41.6 s

Table 5 Complexity of multiquadric-based meshing corresponding to Fig. 12

NOC	AA		BParab	
	NOL	CPU-time	NOL	CPU-time
25	51808	57.44 s	418	0.41 s
49	417346	15 m 38 s	232	0.48 s
100	–	–	316	1.36 s
225	–	–	238	2.30 s
400	–	–	739	12.59 s
625	–	–	736	19.23 s
900	–	–	1147	43.11 s
1156	–	–	1015	48.31s

computed using the AA, BParab and BPBQ-strategies. We extract the zero sets of various RBF-interpolants, and compare the number of leaves (NOL) of the subdivision tree, the number of tetrahedra (NOT) of the spatial subdivision and the CPU-time (CPU). Here CPU-time is the total time for both subdivision and meshing. For all 3D experiments we run our implementations on an Intel 2.4-GHz Pentium 4 machine under Linux with 8-GB RAM, using the g++ compiler, version 3.3.5.

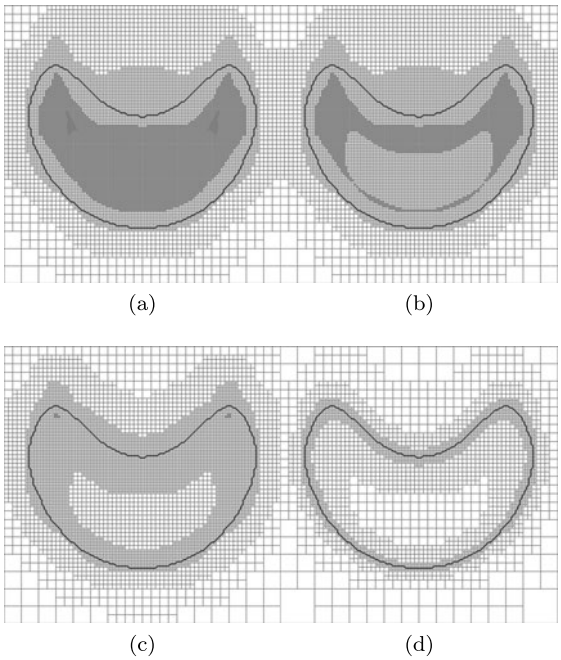


Fig. 13 Isocurve extraction for a cubic RBF-interpolant (100 centers) of the function $f(x, y) = (y - x^2 + 1)^4 + (x^2 + y^2)^4 - 1 = 0$, sampled uniformly on the square $[-1.25, 1.25] \times [-1.5, 1.0]$ using BPBQ without subdivision strategy with (**a**) $SN = 1$, (**b**) $SN = 2^2$, (**c**) $SN = 4^2$, (**d**) $SN = 8^2$

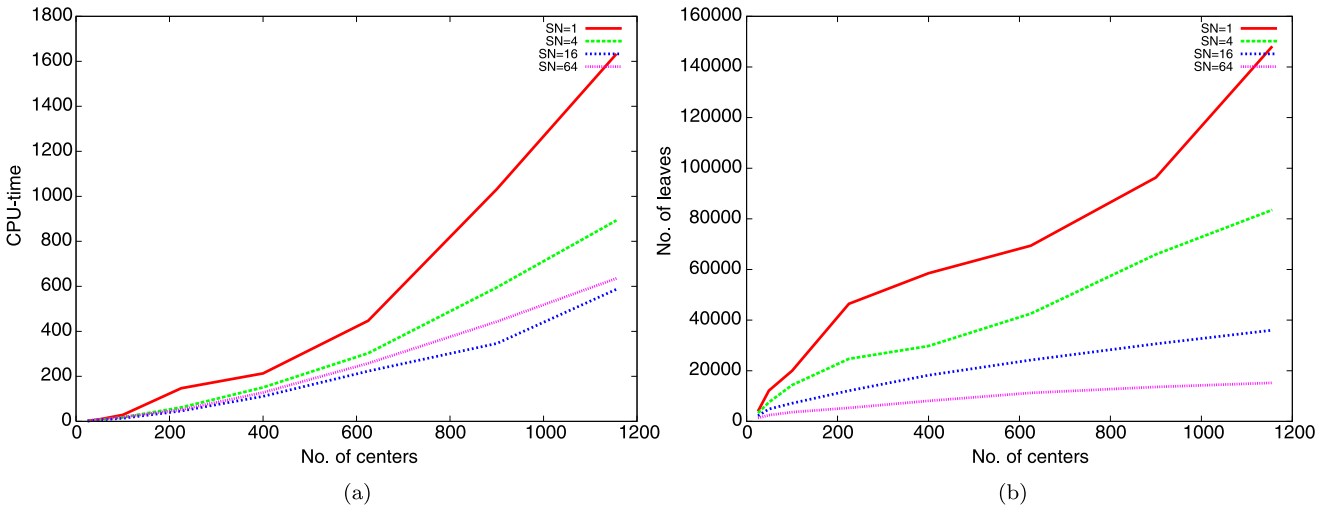
Figure 15 compares the mesh size for different interval computation strategies. It shows that the mesh size using the BParab strategy is coarser than in the case of the BPBQ and AA-strategy.

In Figs. 16, 17, 18, 19, 20 we present some experimental results of isotopic meshing of RBF-based interpolants using BParab strategy. The underlying sample point configurations are described simultaneously.

Off-surface point generation To construct an RBF-interpolant from a point cloud obtained from a smooth surface, one needs to generate good off-surface or offset points. The

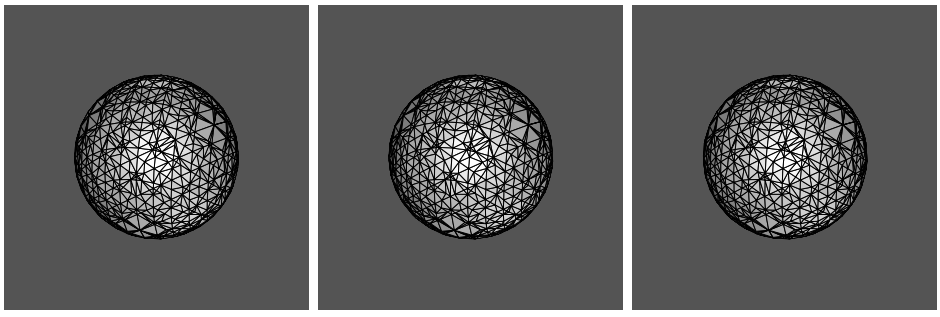
Table 6 Complexity of cubic-based meshing with subdivision-strategy, using different subdivision numbers (*SN*) (corresponding to Fig. 13)

NOC	BPBQ with $SN = 1^2$		BPBQ with $SN = 2^2$		BPBQ with $SN = 4^2$		BPBQ with $SN = 8^2$	
	NOL	CPU-time	NOL	CPU-time	NOL	CPU-time	NOL	CPU-time
25	3994	2.06 s	3286	1.63 s	2188	1.80 s	1318	3.26 s
49	12130	8.36 s	7534	6.53 s	4834	5.95 s	2452	8.40 s
100	19984	28.67 s	14350	17.77 s	7162	13.78 s	3616	19.00 s
225	46435	1 m 47 s	24691	1 m 2 s	12127	46.28 s	5302	51.98 s
400	58516	3 m 33 s	29758	2 m 32 s	18232	1 m 51 s	8086	2 m 8 s
625	69394	7 m 27 s	42556	5 m 3 s	24220	3 m 43 s	11242	4 m 18 s
900	96334	17 m 12 s	65944	9 m 55 s	30634	6 m 46 s	13570	7 m 23 s
1156	148126	27 m 13 s	83536	14 m 53 s	36028	9 m 47 s	15196	10 m 35 s



(a) (b)

Fig. 14 Graphical representation of Table 6: (**a**) number of centers vs. CPU-time, (**b**) number of centers vs. number of leaves of the subdivision tree



(a) (b) (c)

Fig. 15 Isosurface extraction for a cubic-based RBF-interpolant of a point cloud with 252 sample points from a sphere and one offset point near the center of the Sphere inside the box $[-1, 1] \times [-1, 1] \times [-1, 1]$ using strategies: (**a**) AA, (**b**) BPBQ and (**c**) BParab

off-surface point configuration determines the behavior of the RBF-interpolant near the zero level set. In Fig. 21 we consider a point cloud corresponding to the surface of a Head along with its off-surface points. The corresponding RBF-based meshing of the implicit surface is done using the BParab strategy. Figures 22 and 23 show similar experiments with point clouds of a bunny and a horse, respectively.

Tables 7, 8 and 9 compare the space and time complexity for the meshing of RBF-based implicit surfaces (Figs. 16–23) using three main optimization strategies, AA, BPBQ and BParab, respectively. We observe that BPBQ and BParab perform better than AA. Again, between BPBQ and BParab, the time complexity of BParab has better space and time complexity compared to BPBQ.

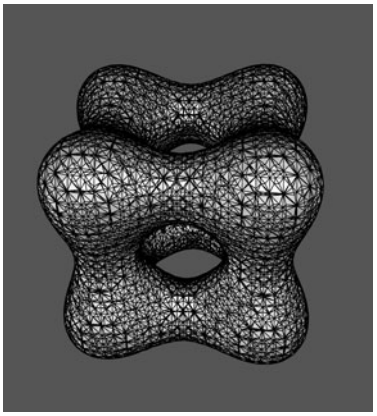


Fig. 16 Tangle (125): RBF-interpolant corresponding to a uniform sampling of the function $f(x, y) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 11$ inside the box $[-3, 3]^3$

Fig. 17 Torus (1211): RBF-interpolant corresponding to 1062 sample points from the surface and 49 points from the medial axis of a torus of radii 8 and 2, respectively, inside the box $[-10, 10]^3$

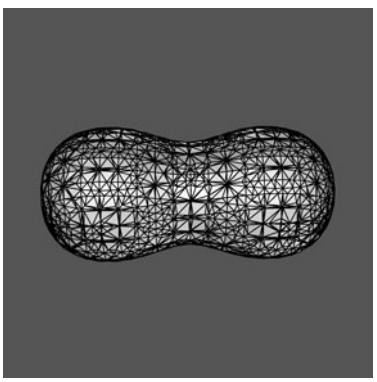


Fig. 18 Blob (343): RBF-interpolant corresponding to a uniform sampling of the function $f(x, y) = \exp(1 - x^2 - (y + \delta)^2 - z^2) + \exp(1 - x^2 - (y - \delta)^2 - z^2] - 1$, $\delta = 0.1$ inside the box $[-4, 4]^3$

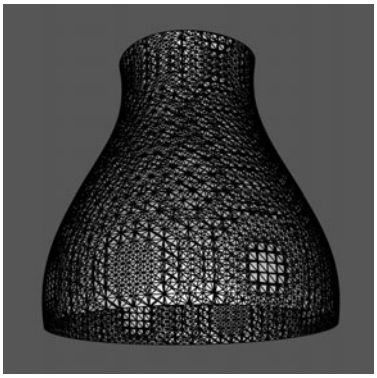


Fig. 19 Potter's Wheel (343): RBF-interpolant corresponding to a uniform sampling of the function $f(x, y) = .5 \sin[x] + \sqrt{y^2 + z^2} - 1.5$ inside the box $[-2, 2]^3$

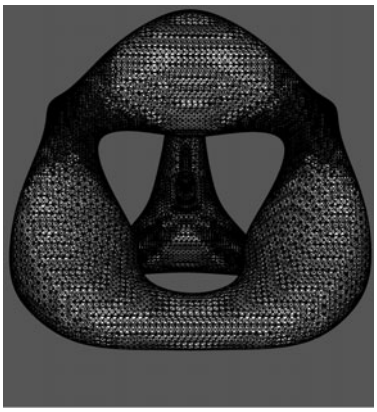


Fig. 20 Chair (343): RBF-interpolant corresponding to a uniform sampling of the function $f(x, y) = (x^2 + y^2 + z^2 - ak^2)^2 - b((z - k)^2 - 2x^2)((z + k)^2 - 2y^2)$, $k = 5$, $a = 0.95$, $b = 0.8$ inside the box $[-5, 5]^3$

5 Conclusion and future work

Our experiments show that IA has unacceptable performance, that AA converges in most experiments for the cubic RBF but fails for the multiquadric-based interpolants. We proposed two strategies for RBF-interpolants, among which BParab is a general and fast method, and the BPBQ-strategy for cubic RBFs gives better results for some experiments than BParab. At this point it is worthwhile to mention some of the future directions of the current work. First, one of the basic requirements for fast convergence of the meshing algorithm is that the input-implicit functions are "well-behaved" (i.e., satisfying a small normal variation condition around their zero-level sets). Constructing such "well-behaved" RBF-interpolants with theoretical guarantees is still an open problem. Secondly, instead of subdividing the whole domain of the implicit function, one could try to find a subset of the domain corresponding to the zero set of the implicit function. This reduces the space and time complexity of the meshing algorithm. Again, we

Table 7 Results using AA strategy

Data (NOC)	NOL	Subdiv.-time
Sphere (253)	848	7.77 s
Tangle Cube (125)	515712	35 m
Potter's Wheel (343)	263964	90 m
Blob (343)	5298392	767 m
Torus (1211)	953261	2828 m
Chair (343)	–	–
Head (429)	–	–
Horse (1308) (Red-Box)	–	–
Bunny (750) (Red-Box)	–	–

Table 8 Results using BPBQ strategy

Data (NOC)	NOL	Subdiv.-time
Sphere (253)	120	0.37 s
Tangle Cube (125)	45221	1 m 49 s
Potter's Wheel (343)	32936	3 m 55 s
Blob (343)	997858	20 m
Torus (1211)	66263	21 m
Chair (343)	735526	71 m
Head (429)	9286523	1209 m
Horse (1308) (Red-Box)	512212	297 m
Bunny (750) (Red-Box)	2632311	753 m

note that $\square s_x(I)\square s_x(I) + \square s_y(I)\square s_y(I)$ is a superset of $\langle \square\nabla s(I), \square\nabla s(I)\rangle$. Therefore, there is room for improving the performance of the meshing algorithm using an improved version of small normal variation condition checking.

Appendix: Function approximation by centered quadratic functions

Our goal is to approximate smooth real-valued functions defined on an interval $[a, b]$ by *centered quadratic functions*, i.e., functions of the form $g(x) = \alpha x^2 + \beta$, i.e., quadratic univariate functions having a critical point at $x = 0$ (if $\alpha \neq 0$). The error is determined with respect to the sup-norm. Our main result is:

Lemma 3 *Consider the function* $f : [a, b] \to \mathbb{R}$, *given by*

$$f(x) = qx^2 + px + r.$$

The best approximant from the class of centered quadratic functions is given by

$$g_*(x) = \alpha_* x^2 + \beta_*,$$

where

$$\alpha_* = \frac{p}{a+b} + q, \quad \textit{and} \quad \beta_* = \frac{a^2 + 6ab + b^2}{8(a+b)}\, p + r.$$

The minimum error is given by

$$E_{\min} := \|f - g_*\|_\infty = \frac{(a-b)^2}{8(a+b)}\,|p|,$$

where $\|\ \|_*$ *denotes the sup-norm on the space of continuous functions on the interval* $I = [a, b]$.

A.1 Approximating Linear functions

First we focus on the approximation of linear functions by centered quadratic functions. More precisely, starting from a linear function $f : [a, b] \to \mathbb{R}$, our goal is to find values of α and β minimizing the expression

$$h(\alpha, \beta) = \max_{x\in[a,b]} \left|\alpha x^2 + \beta - f(x)\right|. \tag{5}$$

To this end, let $f(x) = px$, and consider the function

$$g_{\alpha,\beta}(x) = \alpha x^2 - px + \beta. \tag{6}$$

We may assume that the constant term in f is zero, i.e., $f(0) = 0$, since the constant term can be incorporated in the final value of β. Note that $g_{\alpha,\beta}$ has a critical point at

$$x_c = \frac{p}{2\alpha},$$

assuming $\alpha \neq 0$. We now determine an explicit expression for the error (5), distinguishing the cases (i) $x_c \in [a, b]$ and (ii) $x_c \notin [a, b]$.

A.1.1 (i) $x_c \in [a, b]$

If $x_c \in [a, \frac{1}{2}(a + b)]$, the error is minimal if

$$g_{\alpha,\beta}(b) = -g_{\alpha,\beta}(x_c).$$

Solving β from this equation we get

$$\beta = \beta_*(\alpha) := \frac{1}{2}\left(\frac{p^2}{4\alpha} - b^2\alpha + pb\right). \tag{7}$$

In this case, the error $E(\alpha)$ is

$$E(\alpha) = \left|g_{\alpha,\beta_*(\alpha)}(b)\right| = \frac{1}{8|\alpha|}(2b\alpha - p)^2. \tag{8}$$

If $x_c \in [\frac{1}{2}(a + b), b]$, the error is minimal if

$$g_{\alpha,\beta}(a) = -g_{\alpha,\beta}(x_c).$$

Table 9 Complexity results using BParab strategy

Data (NOC)	NOL	Subdiv.-time	NOL (Balanced)	NOT	Total-time
Sphere (253)	64	0.30 s	64	768	2.30 s
Tangle Cube (125)	34392	56.60 s	34392	475216	9 m
Potter's Wheel (343)	32768	2 m 30 s	32768	393216	26 m
Blob (343)	248298	7 m	248298	2998320	155 m
Torus (1211)	55427	11 m	55700	739826	130 m
Chair (343)	578796	40 m	578796	7704984	412 m
Head (429)	6115628	450 m	6326412	79544486	5012 m
Horse (1308) (inside Red-Box)	320034	100 m	320034	3893060	790 m
Bunny (750) (inside Red-Box)	1456477	276 m	1456477	17755430	2132 m

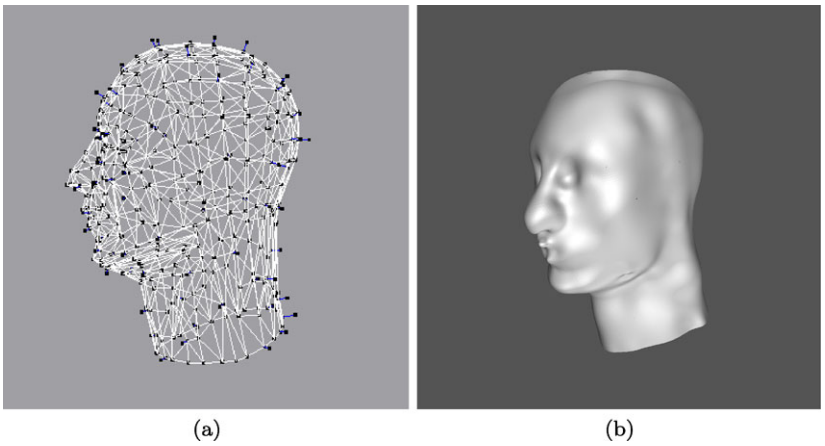


Fig. 21 Head: (**a**) Point cloud with 345 sample points from the surface and 84 off-surface points. (**b**) Meshed implicit surface of the corresponding RBF-interpolant (using BParab strategy)

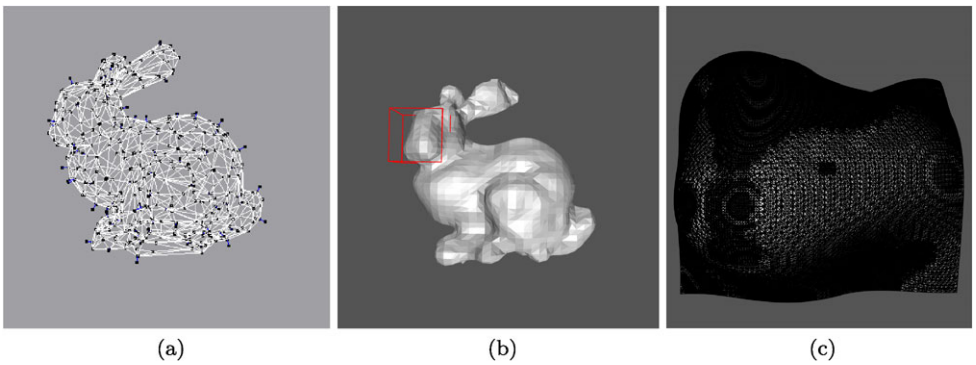


Fig. 22 Bunny: (**a**) Point cloud with 375 sample points from the surface and 375 off-surface points. (**b**) Implicit surface of the corresponding RBF-interpolant visualized using GTS [6]. (**c**) Isotopic meshing of the implicit surface inside the red-box of (**b**) using BParab strategy

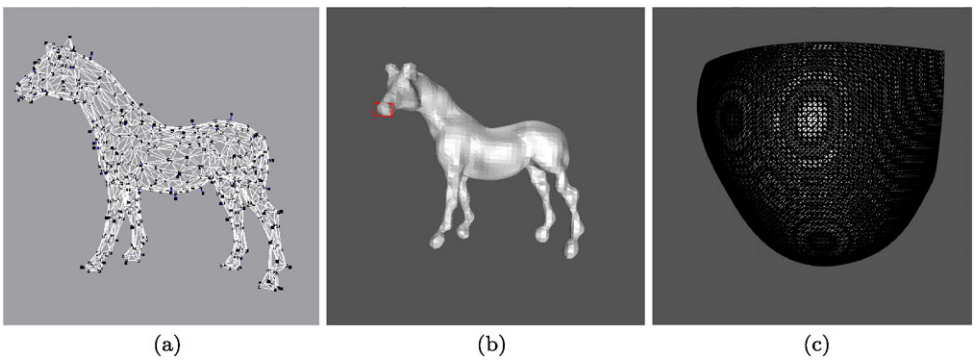


Fig. 23 Horse: (**a**) Point cloud with 436 sample points from the surface and 872 off-surface points. (**b**) Implicit surface of the corresponding RBF-interpolant visualized using GTS [6]. (**c**) Isotopic meshing of the implicit surface inside the red-box of (**b**) using BParab strategy

Solving β from this equation, we get

$$\beta = \beta_*(\alpha) := \frac{1}{2}\left(\frac{p^2}{4\alpha} - a^2\alpha + pa\right). \tag{9}$$

In this case, the error $E(\alpha)$ is

$$E(\alpha) = \left|g_{\alpha,\beta_*(\alpha)}(a)\right| = \frac{1}{8|\alpha|}(2a\alpha - p)^2. \tag{10}$$

A.1.2 (ii) $x_c \notin [a,b]$

In this case, the error is minimal if

$$g_{\alpha,\beta}(a) = -g_{\alpha,\beta}(b).$$

Solving β from this equation we get

$$\beta = \beta_*(\alpha) := \frac{1}{2}\big(p(a+b) - \alpha\big(a^2+b^2\big)\big). \tag{11}$$

In this case, the error $E(\alpha)$ is

$$E(\alpha) = \big|g_{\alpha,\beta_*(\alpha)}(a)\big| = \frac{1}{2}\big(b^2-a^2\big)\left|\alpha - \frac{p}{a+b}\right|. \tag{12}$$

A.2 Minimizing the error

We now determine the minimum of the error function, determined by (8), (10) and (12). First assume that $p > 0$. Then

$$0 < \frac{p}{2b} < \frac{p}{a+b} < \frac{p}{2a}.$$

Furthermore,

$$x_c \in [a,b] \quad \text{iff} \quad \frac{p}{2b} \le \alpha \le \frac{p}{2a}.$$

The analysis of Sects. A.1.1 and A.1.2 shows that

$$E(\alpha) = \begin{cases} \frac{1}{2}(b^2-a^2)(-\alpha + \frac{p}{a+b}), & \text{if } \alpha < \frac{p}{2b}; \\ \frac{(2b\alpha - p)^2}{8\alpha}, & \text{if } \frac{p}{2b} \le \alpha \le \frac{p}{a+b}; \\ \frac{(2a\alpha - p)^2}{8\alpha}, & \text{if } \frac{p}{a+b} \le \alpha \le \frac{p}{2a}; \\ \frac{1}{2}(b^2-a^2)(\alpha - \frac{p}{a+b}), & \text{if } \alpha > \frac{p}{2a}. \end{cases}$$

The graph of this error function is shown in Fig. 24. It is straightforward to check that E is decreasing on $(-\infty, \frac{p}{a+b}]$, and increasing on $[\frac{p}{a+b}, \infty)$. Therefore, the error is minimal if

$$\alpha = \alpha_* := \frac{p}{a+b}.$$

The minimal error is, then,

$$E_{\min} = E(\alpha_*) = \frac{(a-b)^2}{8(a+b)}\,p.$$

Remark 4 Note that the value of β for which the error is minimal is equal to

$$\beta = \beta_* := \frac{(a^2+6ab+b^2)}{8(a+b)}\,p.$$

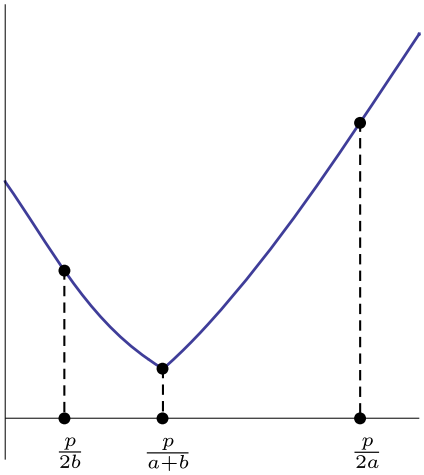


Fig. 24 The error as a function of α, in case $p > 0$

Also in case $p < 0$ the error is minimized for

$$\alpha = \alpha_* := \frac{p}{a+b}, \quad \text{and} \quad \beta = \beta_* := \frac{a^2+6ab+b^2}{8(a+b)}\,p, \tag{13}$$

for which values the error is equal to

$$E_{\min} = \frac{(a-b)^2}{8(a+b)}\,|p|. \tag{14}$$

A.3 Approximating quadratic functions

The analysis of Sect. A.1 can be extended in a straightforward fashion to the approximation of quadratic functions. To see this, let $f(x) = qx^2 + px + r$. It is not hard to see that the minimal error is again given by (14), but the minimum value is attained for $\alpha = \alpha_* + q$ and $\beta = \beta_* + r$, where α_* and β_* are given by (13). Indeed, putting $\overline{\alpha} = \alpha - q$ and $\overline{\beta} = \beta - r$, we see that minimizing the maximum distance between $f(x)$ and $h(x) = \alpha x^2 + \beta$ is equivalent to minimizing the maximum distance between $\overline{f}(x) = px$ and $\overline{h}(x) = \overline{\alpha}x^2 + \overline{\beta}$. This concludes the proof of Lemma 3.

References

1. Boissonnat, J.-D., Cohen-Steiner, D., Vegter, G.: Isotopic implicit surface meshing. Discrete Comput. Geom. **39**, 138–157 (2008)
2. Boost interval arithmetic library: http://www.boost.org (2011)
3. Buhmann, M.: Radial Basis Functions. Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
4. Burr, M., Choi, S., Galehouse, B., Yap, C.: Complete subdivision algorithms, II: Isotopic meshing of singular algebraic curves. In: Proc. Int'l Symp. Symbolic and Algebraic Comp. (ISSAC'08), pp. 87–94, Hagenberg, Austria, Jul. 20–23, 2008 (2008)

5. C++ affine arithmetic library: http://savannah.nongnu.org/projects/libaffa (2011)
6. GTS the gnu triangulated surface library: http://gts.sourceforge.net (2011)
7. de Figueiredo, L.H., Stolfi, J.: Affine arithmetic: concepts and applications. Numer. Algorithms **37**(1–4), 147–158 (2004)
8. Iske, A.: Scattered data modelling using radial basis functions. In: Iske, A., Quak, E., Floater, M.S. (eds.) Tutorials on Multiresolution in Geometric Modelling, Mathematics and Visualization, pp. 287–315. Springer, Heidelberg (2002)
9. Lodha, S., Franke, R.: Scattered data techniques for surfaces. In: Proceedings of Dagstuhl Conference on Scientific Visualization, pp. 182–222. IEEE Computer Society Press, Los Alamitos (1999)
10. Lopes, H., Oliveria, J., Figueiredo, L.: Robust adaptive polygonal approximation of implicit curves. Comput. Graph. **26**(6), 841–852 (2002)
11. Lorensen, W., Cline, H.: Marching cubes: a high resolution 3D surface construction algorithm. Comput. Graph., **21**, 163–169 (1987). Proceedings SIGGRAPH 1987, Annual Conference Series
12. Martin, R., Shou, H., Voiculescu, I., Bowyer, A., Wang, G.: Comparison of interval methods for plotting algebraic curves. Comput. Aided Geom. Des. **19**(7), 553–587 (2002)
13. Moore, R.: Interval Analysis. Prentice-Hall, New York (1996)
14. Plantinga, S., Vegter, G.: Isotopic meshing of implicit surfaces. Vis. Comput. **23**, 45–58 (2007)
15. Powell, M.J.D.: Approximation Theory and Methods. Cambridge University Press, Cambridge (1981)
16. Schaback, R., Wendland, H.: Characterization and construction of radial basis functions. In: Dyn, N., Leviatan, D., Levin, D., Pinkus, A. (eds.) Multivariate Approximation and Applications, Chap. 1, pp. 1–24. Cambridge University Press, Cambridge (2001)
17. Shou, H., Song, W., Shen, J., Martin, R., Wang, G.: A recursive Taylor method for ray casting algebraic surfaces. In: Proc. 2006 Int. Conf. Computer Graphics and Virtual Reality, pp. 196–202. CSREA Press, Las Vegas (2006). ISBN:1932415858
18. Stander, B., Hart, J.: Guaranteeing the topology of an implicit surface polygonizer for interactive modeling. In: Proceedings SIGGRAPH, pp. 279–286 (1997)
19. Stolfi, J., de Figueiredo, L.: Self-validated numerical methods and applications. In: Brazilian Mathematics Colloquium Monograph, IMPA (1997)
20. Wendland, H.: Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics, vol. 17. Cambridge University Press, Cambridge (2004)

A. Chattopadhyay is currently a researcher at the Universite Catholique de Louvain, Belgium. He obtained his Ph.D. in Mathematics from the Johann Bernoulli Institute of Mathematics and Computer Science of the University of Groningen in 2011. His research interests include certified geometric computation, implicit surfaces and visualization.

S. Plantinga is a researcher at the Institute of Mathematics and Computer Science of the University of Groningen, The Netherlands. His research interests include computational geometry and implicit surfaces.

G. Vegter is a professor of Geometry at the Johann Bernoulli Institute for Mathematics and Computer Science at the University of Groningen. He obtained his Ph.D. in Dynamical Systems from the University of Groningen in 1983. His research covers a broad range of subfields of Geometry, ranging from Differential Geometry and Singularity Theory on the more theoretical side, to Computational Geometry and Computer Aided Geometric Design on the more applied side.